Lecture 15, Thurs March 9: Einstein-Certified Randomness

Until recently, the Bell inequality was taught because it was historically and conceptually important, not because it had any practical applications. Sure, it establishes that you can't get away with a local hidden variable theory, but in real life, no one *actually* wants to play the CHSH game, do they? In the last 10 years, however, it's found applications in…

Generating Guaranteed Random Numbers

This is one of the most important tasks in computing (and certainly in cryptography). Once we have quantum mechanics, you might think that the solution is trivial. After all, you can get a random bit by measuring the $|+\rangle$ state in the $\{|0\rangle, |1\rangle\}$ basis. Easy, right? But this solution isn't good enough for cryptography. Cryptographers are paranoid people, and they want the ability to maintain security, even if the hardware they're using was designed by their worst enemy.

These sorts of assumptions aren't just academic speculation, especially given the Snowden revelations. For example, NIST (the National Institute of Standards and Technology) put out a standard for pseudo-random number generation based on elliptic curves to be used for encryption a while back. This standard was later discovered to have a backdoor created by the NSA that would allow them to predict the output numbers, thus being able to break systems encrypted under this standard.

Thus cryptographers want to base their random number generation on the smallest set of assumptions possible. They want bits that are guaranteed to be random, and to be sure that no one added predictability through any sort of backdoor.

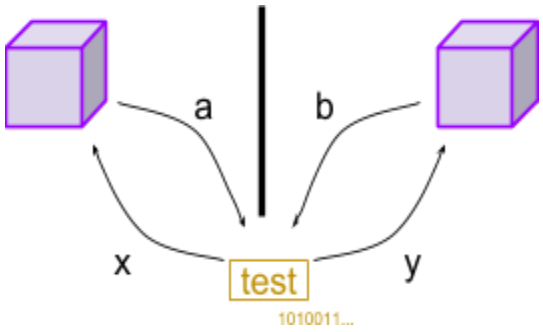
You might think that, logically, one can never prove that numbers are truly random: that the best one can ever say is "*I* can't find any pattern here." After all, you can't prove a negative, and if not the NSA, who's to say that God himself didn't insert a pseudorandom pattern the workings of quantum mechanics?

Though presumably, if God wanted to read our emails he could also do it some other way...

That's what makes it so interesting, and non-obvious, that the Bell inequality *lets us certify numbers as being truly random* under very weak assumptions, which basically boil down to "no faster-than-light travel is possible." Let's now explain how.

Suppose you have two boxes that share quantum entanglement. We'll imagine the boxes were designed by your worst enemy, so you trust nothing about them. All we'll assume is that the boxes can't send signals back and forth (say, because you put them in Faraday cages, or separate them by so large a distance that light has no time to travel between them).

A referee sends the boxes challenge numbers, x and y respectively.
The boxes return numbers a and b respectively.
If the returned numbers pass a test, we'll declare them to be truly random.



So what's the trick? Well, we already saw the trick; it's just the CHSH game!

The usual way to present the CHSH game is as a way for Alice and Bob to prove that they share entanglement—and thus, that the universe is quantum-mechanical, and that local hidden-variable theories are false.

However, winning the CHSH game more than 75% of the time *also* establishes that a and b must have some randomness, that there was some amount of entropy generated.

Why? Because suppose instead that a and b were deterministic functions—i.e., suppose they could be written as $a(x, r)$ and $b(y, r)$ respectively, in terms of Alice and Bob's inputs as well as shared random bits. In that case, whatever these functions were, they'd define a local hidden-variable theory, which is precisely what Bell rules out!

So the conclusion is that, if

(1) x and y are random and

(2) there's no communication between Alice and Bob,

then there must exist at least some randomness in the outputs a and b.

Around 2012, Umesh Vazirani coined the term Einstein-Certified Randomness for this sort of thing. The basic idea goes back earlier—for example, to Roger Colbeck's 2006 PhD thesis, and (in cruder form) to Prof. Aaronson's 2002 review of Stephen Wolfram's *A New Kind of Science*, which used the idea to refute Wolfram's proposal for a deterministic hidden-variable theory underlying quantum mechanics.

OK, so how do we actually extract random bits from the results of the CHSH game?

You could just take the stream of all a's and b's that are output, after many plays of the CHSH game. Admittedly, this need not give us a *uniform* random string. In other words, if the output string x has length n, then its Shannon entropy,

$$\sum p_x \log_2 \frac{1}{p_x}$$

where p_x is the probability of string x, will in general be less than n. However, we can then convert x, if we like, into an (almost) uniformly random string on a smaller number of bits, say $\frac{n}{10}$ or something, by using a well-known tool from classical theoretical computer science called a *randomness extractor*. A randomness extractor—something we already met in the context of quantum key distribution—is a function that crunches down many sort-of-random bits (and, typically, a tiny number of truly random bits, called the *seed*) to fewer very random bits.

David Zuckerman (here at UT) is an expert on randomness extractors.

OK, but there's an obvious problem with this whole scheme.

Namely: we needed the input bits to be uniformly random, in order to play the CHSH game. But that means we put in two perfect random bits, *x* and *y*, in order to get out two bits *a* and *b* that are *not* perfectly random! In other words, the entropy we put in is greater than the entropy we get out, and the whole thing is a net loss.

In his 2006 thesis, Roger Colbeck addressed this by pointing out that you don't have to give Alice and Bob perfectly random bits every time the CHSH game is played. Instead, you can just input $x = y = 0$ most of the time, and occasionally stick in some random *x*'s and *y*'s to prevent Alice and Bob from using hidden variables. Crucially, if Alice or Bob gets a 0 input in a given round, then they have no way of knowing whether that round is for testing or for randomness generation. So, if they want to pass the randomly-inserted tests, then they'll need to play CHSH correctly in *all* the rounds (or almost all of them), which means generating a lot of randomness.

At this point it all comes down to a quantitative question:

So how much entropy can we get out, per bit of entropy that we put in?

There was a race to answer this, by designing better and better protocols that got more and more randomness out per bit of randomness invested. First Colbeck showed how to get cn bits out per n bits in, for some constant $c > 1$. Then Pironio et al. showed how to get $\sim n^2$ bits out per n bits in. Then Vazirani and Vidick showed how to get $\sim \exp(\sqrt{n})$ bits out per n bits in, which is the first time we had exponential randomness expansion. But all this time, an obvious question remained in the background: "why not just use a constant amount of randomness to jumpstart the randomness generation, and then feed the randomness outputted by Alice and Bob back in as input, and so on forever, thereby getting *unlimited* randomness out?"

It turns out that a naïve way of doing this doesn't work: if you just feed Alice and Bob the same random bits that they themselves generated, then they'll *recognize* those bits, so they won't be random *to them*—and that will let Alice and Bob cheat, making their further outputs non-random.

Remember: We're working under the assumption that "Alice" and "Bob" are machines designed by our worst enemy!

If you don't have a limit on the number of devices used, then a simple fix for this problem is to feed Alice and Bob's outputs to two other machines, Charlie and David. Then you can feed Charlie and David's outputs to two more machines, Edith and Fay, and so on forever, getting exponentially more randomness each time.

OK, but what if we have only a *fixed* number of devices (like 4, or 6), and we still want unlimited randomness expansion? In that case, a few years ago Coudron and Yuen, and independently Chung, Miller, Shi, and Wu, figured out how to use the additional devices as "randomness laundering machines"—basically, converting random bits that Alice and Bob can predict into random bits that they *can't* predict, so that then the output bits can be fed back to Alice and Bob for further expansion, and so on as often as desired.

One question that these breakthrough works *didn't* address, was exactly how many random "seed" bits are needed to jump-start this entire process. Like, are we talking a billion bits or 2 bits? In a student project supervised by Prof. Aaronson, Renan Gross calculated the first explicit upper bound, showing that a few tens of thousands of random bits suffice. That's likely still far from the truth: it might be possible with as few as 10 or 20.

It's a pretty amazing conceptual fact that playing the CHSH game can lead to certified random bits (and worth mentioning that this sort of protocol has already been experimentally demonstrated at NIST). So you might wonder…

What else can you certify about two separated boxes, by seeing them win at the CHSH game?

It turns out that the answer is: an *enormous* number of things.

In a tour-de-force in 2012, Reichardt, Unger, and Vazirani showed how to use a sequence of CHSH-like challenges to certify that Alice and Bob performed a specific sequence of unitary transformations on their qubits (up to local changes of basis). This means that, just by making Alice and Bob repeatedly win the CHSH game, you can force them to do *any quantum computation of your choice*. Reichardt et al. describe this as a "classical leash for a quantum system."

This sort of thing constitutes one of the main current ideas for how a classical skeptic could verify the operation of a quantum computer. For (say) factoring a huge integer into primes, we can easily verify the output of a quantum algorithm, by simply multiplying the claimed factors and seeing if they work! But this isn't believed to be the case for all problems that a quantum computer can efficiently solve. Sometimes, the only way to efficiently verify a quantum computer is working correctly might involve using quantum resources yourself. What Reichardt et al. show is that, as long as we have *two* quantum computers, and as long as those quantum computers are entangled with each other but unable to exchange messages, we can use the CHSH game to verify that the computers are behaving as expected.

This brings us nicely to **quantum computation**, which is probably the subject that most of you took the course to learn about! We'll begin discussing quantum computation in earnest in the next lecture.